SYSTEM AND METHOD FOR USING A LEARNING SEQUENCE TO ESTABLISH COMMUNICATIONS ON A HIGH-SPEED NONSYNCHRONOUS INTERFACE IN THE ABSENCE OF CLOCK FORWARDING

TECHNICAL FIELD

[001] The present invention relates to a processor-based system, and more particularly, to a processor-based system having a memory module with a memory hub coupling several memory devices to a processor or other memory access device.

BACKGROUND OF THE INVENTION

[002] Processor-based systems, such as computer systems, use memory devices, such as dynamic random access memory ("DRAM") devices, as system memory to store instructions and data that are accessed by a processor. In a typical computer system, the processor communicates with the system memory through a processor bus and a memory controller. The processor issues a memory request, which includes a memory command, such as a read command, and an address designating the location from which data or instructions are to be read or to which data or instructions are to be written. The memory controller uses the command and address to generate appropriate command signals as well as row and column addresses, which are applied to the system memory. In response to the commands and addresses, data is transferred between the system memory and the processor. The memory controller is often part of a system controller, which also includes bus bridge circuitry for coupling the processor bus to an expansion bus, such as a PCI bus.

[003] Although the operating speed of memory devices has continuously increased, this increase in operating speed has not kept pace with increases in the operating speed of processors. Even slower has been the increase in operating speed of memory controllers coupling processors to memory devices. The relatively slow speed of memory controllers and memory devices limits the data bandwidth between the processor and the memory devices.

[004] One approach to increasing the data bandwidth to and from memory devices is to use multiple memory devices coupled to the processor through a memory hub as shown in Figure 1. A computer system 10 using a memory hub architecture includes a processor 104 for performing various computing functions, such as executing specific software to perform specific calculations or tasks. The processor 104 includes a processor bus 106 that normally includes an address bus, a control bus, and a data bus. The processor bus 106 is typically coupled to cache memory 108, which, is typically static random access memory ("SRAM"). Finally, the processor bus 106 is coupled to a system controller 110, which is also sometimes referred to as a bus bridge.

[005] The system controller 110 contains a memory hub controller 112 that is coupled to the processor 104. The memory hub controller 112 is also coupled to several memory modules 114a-n through a bus system 115. Each of the memory modules 114a-n includes a memory hub 116 coupled to several memory devices 118 through command, address and data buses 117. The memory hub 116 efficiently routes memory requests and responses between the controller 112 and the memory devices 118. Computer systems employing this architecture can have a higher bandwidth because the processor 104 can access one memory module 114a-n while another memory module 114a-n is responding to a prior memory access. For example, the processor 104 can output write data to one of the memory modules 114a-n in the system while another memory module 114a-n in the system is preparing to provide read data to the processor 104. The operating efficiency of computer systems using a memory hub architecture can make it more practical to vastly increase data bandwidth of a memory system. A memory hub architecture can also provide greatly increased memory capacity in computer systems.

[006] The system controller 110 also serves as a communications path to the processor 104 for a variety of other components. More specifically, the system controller 110 includes a graphics port that is typically coupled to a graphics controller 116, which is, in turn, coupled to a video terminal 118. The system controller 110 is also coupled to one or more input devices 120, such as a keyboard or a mouse, to allow an operator to interface with the computer system 10. Typically, the computer system 10 also includes one or more output devices 122, such as a printer, coupled to the processor 104 through the system controller 110. One or more data storage devices 124 are also typically coupled to the processor 104 through the system controller 110 to allow the processor 104 to store data or retrieve data from internal or external storage media (not shown). Examples of typical storage devices 124 include hard and floppy disks, tape cassettes, and compact disk read-only memories (CD-ROMs).

[007] Although there are advantages to utilizing a memory hub for accessing memory devices, the design of the hub memory system, and more generally, computer systems including such a memory hub architecture, becomes increasingly difficult. For example, the memory modules 114a-n each operates internally in a synchronous manner so that the command, address, and data signals transferred to the memory module 114a-n are normally latched or strobed into the memory modules 114a-n by a clock signal. However, operations between memory modules 114a-n are asynchronous. As transfer rates increase, the time during which the command, address and data signals as received at the memory hubs 116 are valid decreases. This period during which the signals are valid is commonly referenced by those ordinarily skilled in the art as the "window" or "eye." Not only does the size of the eye for command, address, and data signals decrease, but the time or location of the eye can also vary because of various factors, such as timing skew, voltage and current drive capability, and the like. In the case of timing skew of signals, it often arises from a variety of timing errors such as loading on the lines of the bus and the physical lengths of such lines.

[008] As the size of signal eyes decrease at higher transfer rates, the variations in the location of the signal eyes become more of a problem. One technique to alleviate this problem to some extent is to couple a clock to the memory modules, a technique known as clock forwarding. As shown in Figure 1, a clock generator 500 generates a clock signal CLK and couples it to the memory hub controller 112 and each of the memory hubs 116 in respective memory modules 114a-n. The memory hubs 116 in respective memory modules 114 a-n also receive command, address and data signals from the memory hub controller 112 that are coupled through the bus system 115. The CLK signal is coupled from the clock generator 500 in synchronism with the command, address and data signals so it, in theory, should be usable by the memory hubs 116 to define the eye during for the command, address and data signals as they are received at the memory hubs 116. However, in practice, even this approach becomes ineffective as signal transfer rates continue to decrease. In particular, the CLK signal may be subject to different conditions than the command, address and data signals, such as being coupled through a physically different signal path or being loaded to a greater degree. Also, for the clock forwarding techniques used in the computer system 10 to successfully function at higher clock speeds, the layout of conductors between the memory hub controller 112 and the memory hubs 116 must be precisely controlled.

[009] One technique that has been proposed to allow the CLK signal to continue being used to strobe command, address and data signals at higher transfer rates is to include circuitry (not shown) in the memory hubs 116 that adjusts the timing of the CLK signal within each of the hubs 116 so that it is aligned with the signal eye. However, this technique adds a fair degree of complexity to the memory hubs 116 and is not always effective.

[010] There is therefore a need for a system and method that allows command, address and data signals to be coupled between a memory hub controller and one or more memory hubs in respective memory modules that avoids problems of synchronizing a clock signal coupled between the memory hub controller and memory hubs along with the command, address, and data signals.

SUMMARY OF THE INVENTION

[011] A memory hub controller is coupled to a memory module having a memory hub and a plurality of memory devices. The memory hub controller communicates with the memory module through an upstream data bus and a downstream data bus. The memory hub controller includes a receiver coupled to the upstream data bus and a transmitter coupled to the downstream data bus. The memory module includes a receiver coupled to the downstream data bus and a transmitter coupled to the upstream data bus. Each of the transmitters is operable in an initialization mode to generate an expected data pattern and to repeatedly couple the generated data pattern to the data bus to which it is coupled. Each of the receivers is operable responsive to a receive clock signal to capture data coupled to the data bus to which it is coupled, including the repeatedly coupled expected data pattern. The receiver being operable in the initialization mode to incrementally alter the phase of the receive clock signal to determine the phases of the receive clock signal that are able to capture received data patterns that match a expected data pattern. The receiver then determines a final value for the phase of the receive clock signal based on the determination of the phases of the receive clock signal that are able to capture received data patterns that match the expected data pattern. This final phase value is then used during normal operation as the phase of the receive clock signal.

BRIEF DESCRIPTION OF THE DRAWINGS

[012] Figure 1 is a block diagram of a computer system that includes several memory modules having a memory hub architecture coupled to a memory hub controller.

[013] Figure 2 is a block diagram of a computer system that includes several memory modules having a memory hub architecture according to one embodiment of the present invention.

[014] Figure 3 is a block diagram of one embodiment of receivers and transmitters used in the computer system of Figure 2 or some other system.

[015] Figure 4 is a block diagram of one embodiment of a pattern comparator used in the receivers of Figure 3.

[016] Figure 5 is a flow chart showing the operation of a receive interface controller that controls the operation of the receivers shown in Figures 3 and 4.

[017] Figure 6 is a block diagram of a memory hub that may be used the memory modules that are used in the computer system of Figure 2.

DETAILED DESCRIPTION OF THE INVENTION

[018] Embodiments of the present invention are directed to a memory module and memory controller each having the capability of generating a clock signal for strobing data signals during the "eye" of the data signals when the data signals are valid. Certain details are set forth below to provide a sufficient understanding of various embodiments of the invention. However, it will be clear to one skilled in the art that the invention may be practiced without these particular details. In other instances, well-known circuits, control signals, and timing protocols have not been shown in detail in order to avoid unnecessarily obscuring the invention. Also, although the embodiments are explained with reference to generating a clock signal to strobe data signals, it will be understood that the same principle can be used to generate a clock signal to strobe command and address signals.

[019] A computer system 100 having a hub memory system according to one embodiment of the invention is shown in Figure 2. The computer system 100 uses many of the same components that are used in the computer system 10 of Figure 1. Therefore, in the interest of brevity, these components have been provided with the same reference numerals, and an explanation of their the functions and operation will not be repeated.

[020] As in the computer system 10 of Figure 1, the system controller 110 also includes a memory hub controller 128 that is coupled to several memory modules 130a,b...n, which serve as system memory for the computer system 100. The memory modules 130 are each coupled to a high-speed downstream bus 132 and a high-speed upstream bus 134. The downstream bus 132 extends downstream from the memory hub controller 128 and downstream from each of the memory modules 130 except the memory module 130n furthest from the memory hub controller 128. Similarly, the upstream bus 134 extends upstream from each of the memory modules 130. Each of these buses 132, 134, include a discrete data bus, although they may also include discrete command and address buses, a combined command/address bus, or some other bus system. However, the explanation of the various embodiments will be with respect to a data bus, it being understood that a similar technique can be used to strobe command and address signals.

[021] The downstream bus 132 couple data away from the memory hub controller 128, and the upstream bus 134 couple data toward the memory hub controller 128. Therefore, the downstream bus 132 couples write data to and from each of the memory modules 130, except for the memory module 130n furthest downstream from the memory hub controller 128, which only receives write data. Similarly, the upstream bus 134 couples read data to and from each of the memory modules 130, except for the memory module 130n furthest downstream from the memory hub controller 128, which only transmits read data. The downstream bus 132 also couples write data from the memory hub controller 128, and the upstream bus 134 couples read data to the memory hub controller 128. Significantly, the buses 132, 134 need not couple clock signals to and from the memory modules 130 and the memory hub controller 128 for the purpose of allowing the memory modules 130 to capture data transmitted through the buses 132, 134. Instead, as explained in greater detail below, each of the memory modules 130 and the memory hub controller 128 generates signals internally to strobe the data coupled through the buses 132, 134.

[022] The memory modules 130 are shown coupled to the memory hub controller 128 in a point-to-point coupling arrangement in which each of the buses 132, 134 are coupled only between two points. However, it will be understood that other topologies may also be used. For example, it may be possible to use a multi-drop arrangement in which a single downstream bus (not shown) and a single upstream bus (not shown) are coupled to all of the memory modules 130. A switching topology may also be used in which the memory hub controller 128 is selectively coupled to each of the memory modules 130 through a switch (not shown). Other topologies that may be used will be apparent to one skilled in the art.

[023] Each of the memory modules 130 includes a first receiver 142 that receives write data through the downstream bus 132, a first transmitter 144 that transmits read data upstream through the upstream bus 134, a second transmitter 146 that transmits write data downstream through the downstream bus 132, and a second receiver 148 that receives read data through the upstream bus 134.

[024] The memory modules 130 also each include a memory hub local 150 that is coupled to its first receiver 142 and its first transmitter 144. The memory hub local 150 receives write data through the downstream bus 132 and the first receiver 142 and couples the write data to one or more of sixteen memory devices 160, which, in the example illustrated in Figure 2, are synchronous dynamic random access memory ("SDRAM") devices. However, a fewer or greater number of memory devices 160 may be used, and memory devices other than SDRAM devices may also be used. The memory hub local 150 is coupled to each of the memory devices 160 through a bus system 164, which normally includes a control bus, an address bus, and a data bus. However, other bus systems, such as a bus system using a shared command/address bus, may also be used.

[025] The memory hub local 150 also receives read data from one or more of the memory devices 160 and couples the read data through the first transmitter 144 and the upstream bus 134. In the event the write data coupled through the downstream bus 132 and the first receiver 142 is not being directed to the memory devices 160 in the memory module 130 receiving the write data, the write data are coupled though a downstream bypass path 170 to the second transmitter 146 for coupling through the downstream bus 132. Similarly, if read data is being transmitted from a downstream memory module 130, the read data is coupled through the upstream bus 134 and the second receiver 148. The read data are then coupled upstream through an upstream bypass path 174, and then through the first transmitter 144 and the upstream bus 134. The second receiver 148 and the second transmitter 146 in the memory module 130n furthest downstream from the memory hub controller 128 are not used and may be omitted from the memory module 130n.

[026] The memory hub controller 128 also includes a transmitter 180 coupled to the downstream bus 132, and a receiver 182 coupled to the upstream bus 134. The downstream bus 132 from the transmitter 180 and the upstream bus 134 to the receiver

182 are coupled only to the memory module 130a that is the furthest upstream to the memory hub controller 128. The transmitter 180 couples write data from the memory hub controller 128, and the receiver 182 couples read data to the memory hub controller 128.

[027] The computer system 100 also includes a reference clock generator 190, which generates a clock signal that is coupled to the memory hub controller 128 and each of the memory modules 130. The memory hub controller 128 and the memory modules 130 use the reference clock to generate two internal clock signals that, in the embodiment of Figure 2, have frequencies of two times, and one-half the frequency of the reference clock signal. The 2x internal clock signal is used as a transmit clock to strobe data from the transmitters 144, 146, 180. As explained in considerable detail below, the receivers 142, 148, 182 adjust the phase of the internal clock signal to generate a receive clock signal that is used to strobe data into the receivers 142, 148, 182. Briefly, the receivers 142, 148, 182 perform this function by receiving a known data pattern from a transmitter 144, 146, 180 to which it is coupled, and attempt to capture that data pattern by strobing the data as the phases of the receive clock signals are incrementally varied. The phase of the receive clock signal that best captures the data pattern is then used to strobe data into the receivers 142, 148, 182 in normal operation.

[028] One embodiment of the receivers 142, 182 and the transmitters 144, 180 in the memory hub controller 128 and in one of the memory modules 130 is shown in Figure 3. In both cases, a receiver 200 functions as both receivers 142, 148 in the memory module 130 and the receiver 182 in the memory hub controller 128, and a transmitter 210 functions as both transmitters 144, 146 in the memory module 130 as well as the single transmitter 180 in the memory hub controller 128. The transmitter 210 includes a pattern generator 220 that generates a predetermined pattern of data bits, and a transmit interface control 224 that controls the transmitting of the pattern. In the embodiment of Figure 3, the same pattern is transmitted on all of the data bits of the buses 132, 134. During an initialization phase of the receiver 200 and transmitter 210, the transmitter 210 continuously repeats its transmission of the data pattern.

[029] As previously explained, the receiver 200 receives the data bits from the transmitter 210 and strobes them in using a receive clock signal generated from the clock signal received from the clock generator 500 and having four times the frequency of the core clock. More specifically, in one embodiment of the invention, the pattern transmitted by the transmitter 210 is the following 32-bit pattern divided into four cycles each having 8 bits: "01011011 11000101 10010011 00101100" (hex "5BC5932C"). The data bit pattern is transmitted from right to left. In the embodiment of Figure 3, a bit is strobed into the receiver 200 on each transition of the receive clock signal, so two bits are captured by the receiver 200 on each receive clock cycle. Since the receive clock has a frequency of four times the core clock, eight bits of data are captured during each cycle of the core clock.

[030] In the embodiment of Figure 3, the first bit is always captured on the positive edge of the receive clock signal. As a result, there are 16 possible patterns of valid data captured by the receiver 200, namely, the transmitted 32-bit pattern shifted by two bits for each pattern. An expected pattern memory 230 stores all 16 of these possible patterns, which, as previously explained, consists of eight bits.

[031] In the embodiment of Figure 3, a pattern comparator 234 performs three comparisons. First, it checks all of the data bits of the bus 132 to ensure that they all have the same value as each data bit is captured since the same data are transmitted on each data bit of the bus 132. The same comparison is performed on the bus 134.

[032] In the second comparison, the pattern comparator 234 compares the eight data bits captured in the receiver 200 for each core cycle to the sixteen valid 8-bit data bit patterns stored in an expected pattern memory 230. For purposes of this comparison, it can use any of the 32 bits captured on each transition of the receive clock signal since the first comparison confirmed that all 32 bits were the same. Based on this comparison, phase adjustment logic 240 adjusts the phase of the receive clock signal so that it can best capture the data coupled to the receiver. More specifically, the pattern comparator 234 compares the 8 bits received during any core cycle to the 16 valid patterns stored in the expected pattern memory 230 to adjust the phase of the receive clock signal. The above operation is controlled by a receive interface controller 244, the operation of which will be explained with reference to the flow chart of Figure 5.

[033] In the third comparison, the pattern comparator 234 checks an additional 33rd bit, which functions as a control bit. The pattern that is sent on the buses 132, 134 is also sent on the control bit for each of these buses. The eight bits captured on one core clock is compared in the same manner as the second comparison.

[034] One embodiment of the pattern comparator 234 is shown in Figure 4 along with the pattern generator 220 and the transmit interface controller 224 in the transmitter 210 and the expected pattern memory 230, the phase adjustment logic 240 and the receive interface controller 244 as shown in Figure 3. The pattern comparator 234 includes a set of 32 double data rate ("DDR") flip-flops 250 that receive the receive clock signal from a receive clock generator 254 and capture 32 bits of data responsive to each transition of the receive clock signal. As each 32 bits of data are captured by the flip-flops 250, the 32 bits of data that were captured on the previous transition of the receive clock signal are transferred to a receive capture buffer 258. The buffer 258 is a recirculating buffer that is able to store data from 24 transitions of the receive clock signal, which occur responsive to twelve periods of the receive clock signal or three periods of the core clock signal. Thus, the buffer 258 stores 768 bits of data (*i.e.*, 24 * 32), and, since it is a recirculating buffer, the oldest data bits stored in the buffer 258 are overwritten with new data bits. The data stored in the receive capture buffer 258 are 32 bits for each of the positive edge and the negative edge of the receive clock signal. There are 12 locations in the buffer 258 that store data for the positive edge and 12 locations in the buffer 258 that store data for the negative edge. Each of these locations is 32 bits wide. The receive capture buffer 258 outputs data from 4 locations for the positive edge and 4 locations for the negative edge. As a result, 256 bits are coupled from the buffer 258, *i.e.*, 32 bits for each of 8 locations. .

[035] The 32 bits from the receive capture buffer 258 are applied to a multiplexer 260, which selects one of four sets of bits for coupling to a set of flip-flops 264. Each set consists of 4 bits from 4 respective locations for the positive edge and 4 bits from 4 respective locations for the negative edge. The first set consists of bits 0, 1, 2, 3 for both the positive and negative edges, the second set consists of bits 4, 5, 6, 7 for both the positive and negative edges, the third set consists of bits 8, 9, 10, 11 for both the positive and negative edges. One of these three sets of eight data bits are selected by a pointer register 266, which is incremented by the receive interface controller 244 in a manner that will be explained below. The flip-flops 264 are clocked by an internal core clock signal that is generated from the reference clock signal.

[036] The eight received data bits captured by the flip-flops 264 are coupled to pattern comparison logic 270, which also receives the sixteen 8-bit patterns stored in the expected pattern memory 230. The pattern comparison logic 270 then issues a pass/fail ("P/F*") signal to the receive interface controller 244 indicative of whether the data bits from the flip-flops 264 match any of the patterns stored in the expected pattern memory 230.

[037] The manner in which the receive interface controller 244 operates the receiver 200 will now be explained with reference to the flow-chart of Figure 5. It will be understood by one skilled in the art that the receive interface controller 244 can be implemented as a properly programmed processor or by some other means.

[038] After the receiver 200 is powered-up, a reset occurs at step 276, an initial startup indicator flag is set to "0" at step 278, and a variable N is set to 0 at step 280. The pattern comparator 234 then determines if the received data pattern is a valid data pattern at step 284. The received pattern will be a valid pattern if the first data bit captured is any even numbered bit, each of which is transmitted on a rising edge of the transmit clock signal. Specifically, if the data pattern "01011011 11000101 10010011 00101100" is transmitted (again, from right to left), a valid data pattern will be any eight-bit sequence of the transmitted pattern that starts on an even bit, *i.e.*, "00101100", "11001011", or "00110010".... If the pattern comparator 134 detected a valid pattern at step 284, it checks the value of the flag at step 286. The flag will initially be the "0" because it was set to that value at step 278. The flag is used to indicate if this is the first pass through step 284. This is needed because an initial passing condition needs to be handled differently from other passes. The pattern comparator 134 will increment a pointer at step 288 to cause the expected pattern memory 230 to output the next 8-bit pattern in sequence, which will subsequently be compared to 8 bits strobed into the receiver 200 by the receive clock signal. Additionally, if the pattern comparator 134 detected a valid pattern at step 284, the phase adjustment logic 240 decrements the phase ("P") of the receive clock signal at step 290 by a number of increments equal to one-half of a receive clock signal period. In the embodiment of Figures 3 and 4, the receive clock signal is divided into 128 increments, so, in the event a valid pattern is detected, the phase of the receive clock signal is decremented by 64 increments. The first pass flag is then set to "1" at step 292, and the operation then returns to step 284, where an invalid pattern should be detected because each data bit that was strobed in by a positive edge of the receive clock signal will now be strobed in by a negative edge of the receive clock signal.

[039] If the pattern comparator 234 detected an invalid pattern at step 284, the phase of the receive clock signal is increased by one increment during step 294, and a check is made at 296 to determine if the phase adjustment causes the phase of the receive clock signal exceeds its limit. If so, the phase of the receive clock signal is reset to an initial value at step 298 and a pointer register 555 is incremented by one. Operation then returns to step 284 to determine if a valid pattern has been received. In summary, if the received data pattern is initially valid, the receive clock is shifted by 180 degrees so that it is no longer valid. When the received pattern either becomes invalid in this manner or is initially invalid, the phase of the receive clock signal is repetitively incremented by 1 by looping through steps 284, 296, and 300.

[040] After steps 284, 296, and 300 have occurred one or more times, the received data pattern will eventually become valid. When this occurs, the "left" edge of the data valid "eye," the minimum phase shift of the receive clock signal that can capture valid data, has been found. The operation then progresses from step 284 to step 286. However, since the flag was set to "1" at either step 292 or step 300, the operation now progress to step 310 where addition phase shifts are added to the receive clock signal to ensure that it will always be able to capture valid data with this phase shift. Specifically, the phase is incremented by 3 increments at step 310, and a determination is made at step 314 whether a variable N that was set to 0 at step 280 is equal to 2. The first time the phase of the receive clock signal is incremented at step 310, N will still be equal to 0. Therefore, the operation will increment the variable in step 318 and return to step 284 to determine if the receive clock signal can still capture valid data. If so, the operation loops through steps 286, 310, 314 and 318 until the variable N is equal to 2. At this point the phase of the receive clock signal is saved at step 320 as the phase P_L corresponding to the left edge of the data valid eye.

[041] After the left edge of the data valid eye has been found, the receive interface controller 244 operates to find the right edge of the data valid eye. It does so by incrementing the phase of the receive clock signal by one increment at step 330 and then checking if doing so causes an invalid data pattern to be captured at step 334. Since the left edge of the data eye was found by the captured data pattern becoming valid, the data pattern is not likely to be invalid during the first pass through step 334. As a result, the operation returns to step 330 to again increment the phase of the receive clock signal. The operation continues to loop through steps 330, 334 until an invalid data pattern is detected at step 334. When this occurs, the "right" edge of the data valid "eye," the maximum phase shift of the receive clock signal that can capture valid data, has been found. The program then saves the phase of the receive clock signal at step 338 as the phase P_R corresponding to the right edge of the data valid eye.

[042] The phase P_F of the receive clock signal that will be used during normal operation is then calculated at step 340 using the formula $P_F=(P_F+P_L)/2$, which sets P_F midway between P_F and P_L. This phase value P_F is then saved at step 344, and normal operation is enabled at step 348.

[043] After the phase P_F of the receive clock signal has been finalized, the receiver 200 in the memory hub controller 128 and each memory module 130 causes its respective transmitter 210 to communicate that fact to an upstream receiver. When the memory hub controller 128 has determined that all of the receivers 200 have been initialized, it ends the initialization mode and begins normal operation. One embodiment of a technique for communicating the synchronization status of the receivers 200 is described in U.S. Patent Application, Serial No. * having a common inventor, which is incorporated herein by reference.

[044] Figure 6 shows an embodiment of the memory hub local 150 according to the present invention, which can be used in the memory modules 130 of Figure 2. The memory hub local 150 include two input bus interfaces 410a,d, which may be used to couple data into the memory hub local 150, and two output bus interfaces 412a,b, which may be used to couple data from the memory hub the memory hub local 150.

[045] The bus interfaces 410a,b, 412a,b are coupled to a switch 460 through a plurality of bus and signal lines, represented by buses 414. The buses 414 are conventional, and include a write data bus and a read data bus, although a single bi-directional data bus may alternatively be provided to couple data in both directions through the bus interfaces 410a,b, 412a,b. It will be appreciated by those ordinarily skilled in the art that the buses 414 are provided by way of example, and that the buses 414 may include fewer or greater signal lines, such as further including a request line and a snoop line, which can be used for maintaining cache coherency.

[046] The switch 460 is coupled to four memory interfaces 470a-d which are, in turn, coupled to the memory devices 160 (Figure 2). By providing a separate and independent memory interface 470a-d for each set of memory devices 160, the memory hub local 150 avoids bus or memory bank conflicts that typically occur with single channel memory architectures. The switch 460 is coupled to each memory interface through a plurality of bus and signal lines, represented by buses 474. The buses 474 include a write data bus, a read data bus, and a request line. However, it will be understood that a single bi-directional data bus may alternatively be used instead of a separate write data bus and read data bus. Moreover, the buses 474 can include a greater or lesser number of signal lines than those previously described.

[047] In an embodiment of the present invention, each memory interface 470a-d is specially adapted to the memory devices 160 to which it is coupled. More specifically, each memory interface 470a-d is specially adapted to provide and receive the specific signals received and generated, respectively, by the memory devices 160 to which it is coupled. Also, the memory interfaces 470a-d are capable of operating with memory devices 160 operating at different clock frequencies. As a result, the memory interfaces 470a-d isolate the processor 104 from changes that may occur at the interface between the memory hub 130 and memory devices 160 coupled to the memory hub local 150, and it provides a more controlled environment to which the memory devices 160 may interface.

[048] The switch 460 coupling the bus interfaces 410a,b, 412a,b and the memory interfaces 470a-d can be any of a variety of conventional or hereinafter developed switches. For example, the switch 460 may be a cross-bar switch that can simultaneously couple bus interfaces 410a,b, 412a,b to each other to provide the downstream bypass path 170 and the upstream bypass path 174 shown in Figure 2. The switch 460 can also be a set of multiplexers that do not provide the same level of connectivity as a cross-bar switch but nevertheless can couple the some or all of the bus interfaces 410a,b, 412a,b to each of the memory interfaces 470a-d. The switch 460 may also includes arbitration logic (not shown) to determine which memory accesses should receive priority over other memory accesses. Bus arbitration performing this function is well known to one skilled in the art.

[049] With further reference to Figure 6, each of the memory interfaces 470a-d includes a respective memory controller 480, a respective write buffer 482, and a respective cache memory unit 484. The memory controller 480 performs the same functions as a conventional memory controller by providing control, address and data signals to the memory devices 160 to which it is coupled and receiving data signals from the memory device 160 to which it is coupled. However, the nature of the signals sent and received by the memory controller 480 will correspond to the nature of the signals that the memory devices 160 are adapted to send and receive. The cache memory unit 484 includes the normal components of a cache memory, including a tag memory, a data memory, a comparator, and the like, as is well known in the art. The memory devices used in the write buffer 482 and the cache memory unit 484 may be either DRAM devices, static random access memory ("SRAM") devices, other types of memory devices, or a combination of all three. Furthermore, any or all of these memory devices as well as the other components used in the cache memory unit 484 may be either embedded or stand-alone devices.

[050] The write buffer 482 in each memory interface 470a-d is used to store write requests while a read request is being serviced. In such a system, the processor 104 can issue a write request to a system memory device 440a-d even if the memory device to which the write request is directed is busy servicing a prior write or read request. The write buffer 482 preferably accumulates several write requests received from the switch 460, which may be interspersed with read requests, and subsequently applies them to each of the memory devices 160 in sequence without any intervening read requests. By pipelining the write requests in this manner, they can be more efficiently processed since delays inherent in read/write turnarounds are avoided. The ability to buffer write requests to allow a read request to be serviced can also greatly reduce memory read latency since read requests can be given first priority regardless of their chronological order.

[051] The use of the cache memory unit 484 in each memory interface 470a-d allows the processor 104 to receive data responsive to a read command directed to a respective system memory device 160 without waiting for the memory device 160 to provide such data in the event that the data was recently read from or written to that memory device 160. The cache memory unit 484 thus reduces the read latency of the system memory devices 440a-d to maximize the memory bandwidth of the computer system. Similarly, the processor 104 can store write data in the cache memory unit 484 and then perform other functions while the memory controller 480 in the same memory interface 470a-d transfers the write data from the cache memory unit 484 to the memory device 160 to which it is coupled.

[052] Further included in the memory hub local 150 may be a self-test module 490 coupled to the switch 460 through a test bus 492. The self-test module 490 is further coupled to a maintenance bus 496, such as a System Management Bus (SMBus) or a maintenance bus according to the Joint Test Action Group (JTAG) and IEEE 1149.1 standards. Both the SMBus and JTAG standards are well known by those ordinarily skilled in the art. Generally, the maintenance bus 496 provides a user access to the self-test module 490 in order to set memory testing parameters and receive test results. For example, the user can couple a separate PC host via the maintenance bus 496 to set the relative timing between signals that are applied to the memory devices 160. Similarly, data indicative of the relative timing between signals that are received from the memory devices 160 can be coupled to the PC host via the maintenance bus 496.

[053] Further included in the memory hub local 150 may be a DMA engine 486 coupled to the switch 460 through a bus 488. The DMA engine 486 enables the memory hub 30 to move blocks of data from one location in one of the memory devices 160 to another location in the memory device without intervention from the processor 104. The bus 488 includes a plurality of conventional bus lines and signal lines, such as address, control, data buses, and the like, for handling data transfers in the system memory. Conventional DMA operations well known by those ordinarily skilled in the art can be implemented by the DMA engine 486.

[054] From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.